AFOSR TR 97-0405

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	~~December 5, 1996~~	2/15/93 - 6/29/96

4. TITLE AND SUBTITLE

~~Deductive Computer Programming~~

5. FUNDING NUMBERS

C: F49620-93-1-0139

6. AUTHOR(S)

Zohar Manna

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Computer Science Department
Gates Building - 4B
Stanford University
Stanford, CA 94305-9045

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFOSR/NM
110 Duncan Avenue, Room B115
Bolling AFB, DC 20332-8080

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

19971006 041

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The REACT research group at Stanford, under the supervision of Professor Zohar Manna, developed methodologies and tools for the verification and synthesis of reactive, real-time and hybrid systems based on their temporal specifications. A system, STeP (Stanford Temporal Prover), has been implemented to support computer-aided verification and synthesis based on these methodologies and tools. The goal of the system is to automate the development process as much as possible, thereby reducing the errors that otherwise pervade software development.

The research group consisted of Prof. Zohar Manna (PI), Prof. Amir Pnueli (visitor), 8 PhD students, 2 MSc students, and a programmer. One of the PhD students graduated during the period covered by this report. Several of the PhD students were supported by this AFOSR grant.

14. SUBJECT TERMS	15. NUMBER IF PAGES
	11
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TEMPORAL VERIFICATION AND DEVELOPMENT OF REACTIVE PROGRAMS

Final Report:
Department of the Air Force
Award Number: F49620-93-1-0139

by
Zohar Manna, Professor
Computer Science Department
Stanford University
Stanford, California 94305

November 1996

SUMMARY OF PROGRESS

The REACT research group at Stanford, under the supervision of Professor Zohar Manna, developed methodologies and tools for the verification and synthesis of reactive, real-time and hybrid systems based on their temporal specifications. A system, STeP (Stanford Temporal Prover), has been implemented to support computer-aided verification and synthesis based on these methodologies and tools. The goal of the system is to automate the development process as much as possible, thereby reducing the errors that otherwise pervade software development.

The research group consisted of Prof. Zohar Manna (PI), Prof. Amir Pnueli (visitor), 8 PhD students, 2 MSc students, and a programmer. One of the PhD students graduated during the period covered by this report. Several of the PhD students were supported by this AFOSR grant.

Research has been conducted on the following related areas:

- **Methodologies and Tools for Computer-Aided Verification and Synthesis**

  Reactive Systems A methodology based on temporal logic was developed for the specification, verification and synthesis of reactive systems, i.e., systems whose role is to maintain an ongoing interaction with their environment. Examples of reactive systems are concurrent and embedded programs, communication networks, control programs for industrial plants and transport systems, and operating systems.

  Real-Time Systems The temporal methodology has been extended to deal with real-time systems by introducing a time parameter into the state. The specification language and the proof system were appropriately extended to allow the expression and verification of timing properties of such systems.

  Hybrid Systems The temporal methodology, the specification language, and the proof system have been extended to deal with hybrid systems. These systems consist of a non-trivial mixture of discrete and continuous components.

- **The Stanford Temporal Prover (STeP) System**

We are currently implementing a comprehensive development environment that is particularly suited for the construction of highly reliable software, because of methodology and design features that are not available in other systems for formal reasoning. This includes

Compositional Reasoning Facilities for Compositional Reasoning (Modularity) are incorporated in STeP, which allows large systems to be analyzed piecemeal and promotes software reuse of formally verified system components.

Integration of Methods The system is unique in its integration of fundamental principles of mathematical deduction, model checking, decision procedures, and verification rules. Each of these technologies can be applied, as appropriate, to the system being developed.

Visual Language An important special feature of the STeP system is its use of Verification Diagrams as a visual language for constructing proofs. We expect that this language will be a significant improvement towards allowing software engineers who have not been trained in formal methods to apply these tools to system development.

TECHNICAL INFORMATION

During the second year of the award, our research concentrated on the following topics:

REACTIVE SYSTEMS

- **Verification Diagrams** ([MP1,BMS])

Most formal approaches to the verification of temporal properties of reactive programs infer temporal conclusions from verification conditions that are state formulas, i.e., contain no temporal operators. These proofs can often be effectively presented by the use of *verification diagrams*. We introduced verification diagrams as a tool for proving various temporal properties.

Beginning with safety properties, we present *wait-for* and *invariance* diagrams for proving wait-for (precedence) and invariance formulas. Proceeding to liveness properties, we present verification diagrams for response properties that require a bounded number of helpful steps (*chain* diagrams) and response properties that require an unbounded number of helpful steps (*rank* diagrams).

Additional types of diagrams are proposed for handling response properties for parameterized programs and response properties that rely on the full spectrum of fairness requirements, including compassionate helpful transitions.

- **Realizability and Synthesis** ([AM])

We present two algorithms: a realizability-checking algorithm and a synthesis algorithm. Given a specification of reactive asynchronous modules expressed in propositional ETL (Extended Temporal Logic), the realizability-checking algorithm decides whether the specification has an actual implementation, under the assumptions of a random environment and fair execution. It also creates a structure which can then be transformed by the synthesis algorithm into a program, represented as a labeled finite automaton. Unlike previous approaches, the realizability-checking algorithm can handle fairness assumptions. The realizability-checking algorithm is incremental and it directly manipulates formulas in linear temporal logic without having to transform into a branching-time logic or other representations.

- **Verification of Parameterized Programs** ([MP2])

We developed special techniques for the verification of parameterized reactive programs, using temporal logic. A parameterized program consists of several similar processes whose number is determined by an input parameter. A challenging problem is to provide methods for the *uniform* verification of such programs, i.e., proving correctness of the program for *any* number of processes. The ability to conduct a uniform verification of a parameterized program is one of the striking advantages of the deductive method for temporal verification over model-checking techniques.

Suppose M denote the input parameter which determines the number of processes in the considered system. We can use model-checking to verify the desired properties of the system for specific values of M, such as $M = 3, 4, 5$. Usually, the model checker's memory capacity is exceeded for values of M smaller than 100. Furthermore, in the general case, nothing can be concluded about the property holding for any value of M from the fact that it holds for some finite set of values. In comparison, the deductive method establishes in one fell swoop the validity of the property for any value of M.

The verification approach presented here extends our methodology, proof rules and verification diagrams to the case of parameterized programs. We applied our techniques in the verification of of safety properties of complex parameterized programs.

- **Temporal Verification of Simulation and Refinement** ([KMP])

We developed temporal logic methods for proving simulation and refinement relations between programs. We first defined the relations of simulation and refinement between programs and relate them to inclusion relations between computations and observations of the compared systems. These semantic definitions can be formalized in temporal logic by the use of the temporal and observational semantics formulas. This representation expresses simulation and refinement as implications between a pair of such formulas. We provided proof rules.

We also proposed a new temporal logic, called TLR, which is insensitive to stuttering. This logic is interpreted over sequences of sampling points, alternating between persistent and transient sample points. This logic possesses an idempotent next-time operator, which gives some insight into its stuttering robustness. We gave a decision procedure and a complete axiomatic system for the propositional version of TLR.

We also developed a stronger proof rule for refinement, and illustrated its use to prove refinement of two programs that could not be done within the regular temporal logic framework.

REAL-TIME SYSTEMS

- **Verification of Real-Time Systems** ([HMP,MP3])

We extend the specification language of temporal logic, the corresponding verification framework, and the underlying computational model to deal with real-time properties of reactive systems. The abstract notion of timed transition systems generalizes traditional transition systems conservatively: qualitative fairness requirements are replaced (and superseded) by quantitative lower-bound and upper-bound timing constraints on transitions. This framework can model real-time systems that communicate either through shared variables or by message passing and handle real-time issues such as timeouts, process priorities (interrupts), and process scheduling.

We exhibit two styles for the specification of real-time systems. While the first approach uses time-bounded versions of the temporal operators, the second approach allows explicit references to time through a special clock variable. Corresponding to the two styles of specification, we present and compare two different proof methodologies for the verification of timing requirements that are expressed in these styles. For the *bounded-operator* style, we provide a set of proof rules for establishing bounded-invariance and bounded-response properties of timed transition systems. This approach generalizes the standard temporal proof rules for verifying invariance and response properties conservatively. For the *explicit-clock* style, we exploit the observation that every time-bounded property is a safety property and use the standard temporal proof rules for establishing safety properties.

- **Compositional Verification of Real-time Systems** ([CMP])

We developed a compositional proof system for the verification of real-time systems. Real-time systems are modeled as timed transition modules, which explicitly model interaction with the environment and may be combined using composition operators.

Composition rules are devised such that the correctness of a system may be determined from the correctness of its components. These proof rules are demonstrated on Fischer's mutual exclusion algorithm, for which mutual exclusion and bounded response are proven.

HYBRID SYSTEMS

- **Hybrid Temporal Logic** ([KHMP])

We propose a methodology for the specification, verification, and design of hybrid systems. The methodology consists of the computational model of *Concrete Phase Transition Systems* (CPTS), the specification language of *Hybrid Temporal Logic* (HTL), the graphical system description language of *Hybrid Automata*, and a proof system for verifying that hybrid automata satisfy their HTL specifications.

The novelty of the approach lies in the continuous-time logic, which allows specification of both point-based and interval-based properties and provides direct references to derivatives of variables, and in the proof system that supports verification of point-based and interval-based properties. The proof rules demonstrate that sound and convenient induction rules can also be established for continuous-time logics and are not necessarily restricted to discrete logics. The proof rules are illustrated on several examples.

- **Design of Controlled Systems** ([SM,MS])

We propose a conceptual framework to support specification, design and verification of programs controlling physical systems. We introduce a computational model that represents the controller capabilities and distinguishes between synchronous and phase transitions. A graphical system description language is proposed that we believe is readily accessible to control engineers. We formalize the notion of control strategy in controller design.

- **Verification in Continuous Time** ([dAM])

There are two common choices for the semantics of real-time and hybrid systems. The first is a *discrete semantics*, in which the temporal evolution of the system is represented as an enumerable sequence of snapshots, each describing the state of the system at a certain time. The second is a *continuous semantics*, in which the system evolution is represented by a sequence of intervals of time, together with a description of the system state during each interval.

We showed how the advantages of both semantics can be combined by adapting the simple verification rules of the discrete semantics to the continuous one. Specifically, if a temporal logic formula has the property of *finite variability* (FV), its validity in the discrete semantics implies its validity in the continuous one. Thus, verification rules that have as conclusion a FV formula are sound also in the continuous semantics. Most formulas that arise in practice are FV, and we give criteria that help to characterize them. The ability to transfer results between the semantics, together with a deductive system for the continuous semantics, provides a methodology for verifying temporal logic properties of real-time and hybrid systems in the continuous semantics.

COMPUTER-AIDED VERIFICATION

- **STeP: the Stanford Temporal Prover** ([M,MBB1,MBB2])

STeP (the Stanford Temporal Prover) is a system to support computer-aided verification of reactive systems based on their temporal specifications. Unlike most systems for temporal verification, STeP does not concentrate solely on finite-state systems. It combines model checking with algorithmic deductive methods (decision procedures) and interactive deductive methods (theorem proving). The user is expected to interact with the system and provide, whenever necessary, top-level guidance in the form of auxiliary invariants for safety properties, and well-founded measures and intermediate assertions for progress properties. In short, STeP has been designed with the objective of combining the expressiveness of deductive methods with the simplicity of model checking.

Development efforts have been focused, in particular, in the following areas.

First, in addition to the textual language of temporal logic, the system supports a structured visual language of *verification diagrams* [MP1] for guiding proofs. Verification diagrams allow the user to construct proofs hierarchically, starting from a high-level, intuitive proof sketch and proceeding incrementally, as necessary, through layers of greater detail.

Second, the system implements powerful techniques (algorithmic and heuristic) for automatic *invariant generation.* Deductive verification in the temporal framework almost always relies heavily on finding, for a given program and specification, suitably strong invariants and intermediate assertions. The user can typically provide an intuitive, high-level invariant, from which the system derives stronger, *top-down invariants.* Simultaneously, *bottom-up invariants* are generated automatically by analyzing the program text. By combining these two methods, the system can often deduce sufficiently detailed invariants to carry through the entire verification process.

Finally, the system provides a built-in facility for automatically checking a large class of first-order and temporal formulas, based on simplification methods, term rewriting, and decision procedures. This degree of automated deduction is sufficient to handle most of the verification conditions that arise during the course of deductive verification — and the few conditions that are not solved automatically correspond to the critical steps of manually constructed proofs, where the user is most capable of providing guidance.

Although the system is in an early stage of development, many of the examples in the Manna-Pnueli textbook [MP3] have already been automatically verified using STeP. The system is already used as a tool for doctoral research and about to be released worldwide as courseware.

- **Automatic Generation of Invariants** [BBM]

Verifying temporal specifications of reactive and concurrent systems commonly relies on finding auxiliary invariants and strengthening properties. Two dual approaches find solutions to these problems: the *bottom-up method* performs an abstract execution of the system, the *top-down method* performs a reverse execution. Both methods are impractical for very large scale verification. We developed new techniques, applying abstract interpretation, to obtain feasible solutions even for complex tasks.

We apply tools from linear algebra, linear programming, (tree-)automata theory and theorem proving. Perpendicular to the techniques, we distinguish abstraction domains by their ability to contain mutual dependencies of variables. The new techniques have been implemented in STeP, the Stanford Temporal Prover.

- **Binary Decision Diagrams** [AMU]

We introduce a class of Binary Decision Diagrams (BDDs) which we call Differential BDDs (ΔBDDs), and two transformations over ΔBDDs, called Push-up and Delta transformations. In ΔBDDs and its derived classes, in addition to the ordinary node-sharing in the normal Ordered Binary Decision Diagrams (OBDDs), some isomorphic substructures are collapsed together forming an even more compact representation of boolean functions.

The elimination of isomorphic substructures coincides with the repetitive occurrences of the same or similar small components in many applications of BDDs such as in the representation of hardware circuits. The reduction in the number of nodes, from OBDDs to ΔBDDs, is potentially exponential while boolean manipulations on ΔBDDs remain efficient.

- **Temporal Deduction** ([MMW])

We developed a deductive system for predicate temporal logic with induction. This deductive system is relatively complete.

Representing temporal operators by first-order expressions enables temporal deduction to use the already developed techniques of first-order deduction. But when translating from temporal logic to first-order logic is done indiscriminately, the ensuing quantifications and comparisons of time expressions encumber formulas, hindering deduction. In our deductive system, translation occurs more carefully, via *reification* rules. These rules paraphrase selected temporal formulas as nontemporal first-order formulas with *time annotations*. This process of time reification suppresses quantifications (the process is analogous to quantifier skolemization) and uses addition instead of complicated combinations of comparisons. Some ordering conditions on arithmetic expressions can arise, but these are handled automatically by a special-purpose unification algorithm plus a decision procedure for Presburger arithmetic.

- **Algorithmic-Deductive Verification** ([SUM,dAM2])

The two main approaches to verifying temporal properties of hardware are Algorithmic (Model Checking) and Deductive (Theorem Proving) . Algorithmic verification is usually automatic, while deductive verification often relies on user interaction to identify suitable lemmas and auxiliary assertions. However, the algorithmic approach is usually only applicable to purely finite-state systems of limited size, while the deductive approach can verify infinite-state systems and parameterized finite-state systems of arbitrary size.

While algorithmic methods can automatically verify a particular hardware configuration for a given maximum number of bits, determined by the computational resources available, deductive methods have the potential of verifying the corresponding parameterized class of systems in a single, interactive step. We investigated how a combination of the expressiveness of deduction with algorithmic nature of model checking can enable the verification of larger and more complex systems than what either approach can allow.

To manage the complexity of such verification efforts, we developed Diagram-Based verification formalisms. These formalisms offer concise, high-level representations of formal proofs, and can be developed in a hierarchical and incremental way. Diagrams can also be easier to construct, understand and maintain than text-based proofs and specifications, and can be incorporated into the hardware design process.

PH.D. THESIS: Temporal Validity [Mc]

This dissertation presents two methods for determining satisfiability or validity of formulas of Discrete Metric Annotated Linear Temporal Logic. This logic is convenient for representing and verifying properties of reactive and concurrent systems, including software and electronic circuits.

The first method is an algorithm for automatically deciding whether any given propositional temporal formula is satisfiable, and if so reporting a model of the formula. The classical algorithm for this task defines a possible state as a setting of the truth-values of certain formulas which are relevant to the given formula; possible states are constructed and then linked according to their associated formulas' constraints on temporally adjacent states, and then some fulfillment-conditions are checked. The new algorithm here efficiently extends that treatment to formulas with operators which refer to the past or are metric (i.e. refer to measured amounts of time). Then, whereas classical proofs of correctness for such algorithms are existential, the proof here is constructive. It shows that for any given formula being checked, any model of the formula is embedded in the graph of possible states, in which case the algorithm here can find the model.

The second method is a deduction-calculus for determining the validity of predicate temporal formulas. As has been done, to use the already well-developed techniques available for deduction in first-order logic, it is useful to represent temporal operators by first-order expressions, with time reified as expressions of the natural numbers. The new deduction-calculus presented here employs a refined, conservative version of the process of translation from temporal forms to expressions with time reified. Quantifications are elided, and addition is used instead of classical complicated combinations of comparisons. Some ordering conditions on arithmetic expressions can arise, but such are handled automatically via unification and a decision-procedure for Presburger arithmetic. With deduction-rules such as temporal induction, this deduction-calculus is as powerful as others. Further deduction-rules such as rewriting are included for additional convenience.

BOOK: Temporal Verification [MP4]

The book *Temporal Verification of Reactive Systems* [MP3] describes formal methods for the verification of reactive systems. The book is already being used as course text in various universities.

We studied in detail the proof methodologies for verifying temporal properties of reactive systems. Appropriate proof principles were presented for each class of temporal *safety* and *progress* properties.

We developed proof principles for the establishment of *safety* properties. We showed that essentially there is only one such principle for safety proofs, the invariance principle, which is a generalization of the method of intermediate assertions. We also indicated special cases under which these assertions can be found algorithmically.

The proof principle that we developed for *liveness* properties is based on the notion of well-founded descent of ranking functions. However, because of the nondeterminacy inherent in concurrent computations, the well-founded principle must be modified in a way that is strongly dependent on the notion of *fairness* that is assumed in the computation. Consequently, three versions of the well-founded principle were presented, each corresponding to a different definition of fairness.

We illustrate the application of these rules on many examples. We suggest concise presentations of complex proofs using the devices of *transition tables* and *verification diagrams.*

PUBLICATIONS

Research Papers, PhD Thesis, Software, and (research behind the) Book, mentioned in this list, were partially supported by the AFOSR grant.

INVITED PAPERS

[M] Z. Manna, "Beyond Model Checking," 6th Conference on *Computer Aided Verification*, Lecture Notes in Computer Science 818, Springer-Verlag, 1994. pp. 220–221.

[MP1] Z. Manna and A. Pnueli, "Temporal Verification Diagrams," International Symposium on *Theoretical Aspects of Computer Software*, Lecture Notes in Computer Science 789, Springer-Verlag, 1994, pp. 726–765.

[KMP] Y. Kesten, Z. Manna, and A. Pnueli, "Temporal Verification of Simulation and Refinement," REX Symposium *A Decade of Concurrency*, Lecture Notes in Computer Science 803, Springer-Verlag, 1994, pp. 273–346.

[MP2] Z. Manna and A. Pnueli, "Verification of Parameterized Programs," In *Specification and Validation Methods*, (E. Borger, ed.), Oxford University Press, 1994, pp. 167–230.

[AMU] A. Anuchitanukul, Z. Manna, and T. Uribe, "Differential BDDS," Lecture Notes in Computer Science 1000, Springer-Verlag, 1995, pp. 218–233.

[BBM] N. Bjorner, A. Browne, and Z. Manna, "Automatic Generation of Invariants and Intermediate Assertions," 1st International Conference on Principles and Practice of Constraint Programming, Cassis, France. Lecture Notes in Computer Science 976, 1995, pp. 589–623.

[MP3] Z. Manna and A. Pnueli, "Clocked Transition Systems," A tribute to Prof. C.S.Tang on his 70th birthday, Beijing, China, August 1995.

CONFERENCE PAPERS

[KHMP] A. Kapur, T. Henzinger, Z. Manna, and A. Pnueli, "Proving Properties of Hybrid Systems," International Symposium on *Formal Techniques in Real Time and Fault Tolerant Systems*, Lecture Notes in Computer Science 863, Springer-Verlag, 1994, pp. 431–454.

[AM] A. Anuchitanukul and Z. Manna, "Realizability and Synthesis of Reactive Modules," 6th International Conference on *Computer Aided Verification*, Lecture Notes in Computer Science 818, Springer-Verlag, 1994, pp. 156–168.

[CMP] E. Chang, Z. Manna and A. Pnueli, "Compositional Verification of Real-Time Systems," IEEE Symposium on *Logic in Computer Science*, Paris, Lecture Notes in Computer Science, Springer-Verlag, 1994, pp. 458–465.

[MMW] H. McGuire, Z. Manna and R. Waldinger, "Annotation-Based Deduction in Temporal logic," First International Conference on Temporal Logic, Lecture Notes in Computer Science 827, Springer-Verlag, 1994. pp. 430–444.

[SM] H.B. Sipma and Z. Manna, "Specification and Verification of Controlled Systems," International Symposium on *Formal Techniques in Real Time and Fault Tolerant Systems*, Lecture Notes in Computer Science 863, Springer-Verlag, 1994, pp. 641–659.

[dAM1] L. de Alfaro and Z. Manna, "Continuous Verification by Discrete Reasoning," 4th International Conference on Algebraic Methodology and Software Technology (AMAST), Montreal, Canada, Lecture Motes in Computer Science 936, Springer-Verlag, July 1995, pp. 292–306.

[MS] Z. Manna and H.B. Sipma, "A Deductive Approach Towards Controller Synthesis," 10th IEEE International Symposium on Intelligent Control, IEEE Computer Society Press, Monterey, CA, August 1995, pp. 35–41.

[BMS] A. Browne, Z. Manna, and H. Sipma, "Generalized Verification Diagrams," Foundations of Software Technology and Theoretical Computer Science (Bangalore, India). Lecture Notes in Computer Science 1026, Springer-Verlag, pp. 484–498, December 1995.

[SUM] H.B. Sipma, T.E. Uribe, and Z. Manna, "Deductive Model Checking," 8th International Conference on *Computer Aided Verification*, Lecture Notes in Computer Science 1102, Springer-Verlag, 1996, pp. 208–219.

[dAM2] L. de Alfaro and Z. Manna, "Temporal Verification by Diagram Transformations," 8th International Conference on *Computer Aided Verification*, Lecture Notes in Computer Science 1102, Springer-Verlag, 1996, pp. 288–299.

JOURNAL ARTICLES

[HMP] T. Henzinger, Z. Manna, and A. Pnueli, "Temporal Proof Methodologies for Timed Transition Systems," Information and Computation journal, Vol. 112, No. 2, 1994, pp. 273–337.

Ph.D. THESES

Supervised by Zohar Manna.

[Mc] H. McGuire, *Two Methods for Checking Formulas of Temporal Logic,* Computer Science Department, Stanford University, 1995.

BOOKS

[MP4] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Safety*, Springer-Verlag, New York (June 1995).

SOFTWARE

[MBB1] Z. Manna, N. Bjorner, A. Browne, E. Chang, M. Colon, L. de Alfaro, H. Devarajan, H. Sipma, and T. Uribe, *STeP: Stanford Temporal Prover*, Computer Science Department, Stanford University, 1994.

[MBB2] Z. Manna, N. Bjorner, A. Browne, E. Chang, M. Colon, L. de Alfaro, H. Devarajan, A. Kapur, J. Lee, H. Sipma, and T. Uribe, "STeP: The Stanford Temporal Prover," TAPSOFT'95: Theory and Practice of Software Vevelopment, Aarhus, Denmark, Lecture Notes in Computer Science 915, Springer-Verlag, May 1995, pp. 793–794.

[BBC] N. Bjorner, A. Browne, E. Chang, M. Colon, A. Kapur, Z. Manna, H.B. Sipma, and T. Uribe, "STeP: Deductive-Algorithmic Verification of Reactive and Real-time Systems," 8th International Conference on *Computer Aided Verification*, Lecture Notes in Computer Science 1102, Springer-Verlag, 1996, pp. 415–418.

PERSONNEL

STUDENTS INVOLVED IN RESEARCH

Hugh McGuire (graduated)
Luca De Alfaro (5th year)
Arjun Kapur (4th year, has NSF Fellowship)
Tomas Uribe (4th year)
Anuchit Anuchitanukul (graduated)
Henny Sipma (4th year)
Nikolaj Bjorner (2nd year)
Michael Colon (1st year)
Jeffrey Kamerer (MSc student)
Hsiao-Lan Liao (MSc student)

COLLABORATION

Amir Pnueli (Professor of Computer Science at the Weizmann Institute and Visiting Scientist at Stanford University)

Zeev Shtadler (Senior Scientist, Logic CAD group, INTEL)

Richard Waldinger (Principal Scientist at SRI International and Consulting Professor at Stanford University)

Tom Henzinger (Professor of Computer Science at University of California, Berkeley)